Name: Probin Bhagchandani

ID: 22DCE006

Subject: OCCSE4002: Social Network Analysis

Department: DEPSTAR Computer Engineering (DCE)

External Practical Exam

Practical-1)

Aim: During a tech event, you are tasked with analyzing Instagram interactions. You must authenticate with the Instagram Graph API, fetch posts using specific hashtags, and build a user interaction graph based on comments and likes.

Implementation Code:

```
#Practical-1
#Instagram interaction graph

!pip install networkx matplotlib pandas numpy python-louvain seaborn

import random
import string
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
from community import community_louvain

def random_username(length=8):
    return ''.join(random.choices(string.ascii_lowercase +
string.digits, k=length))

def generate_synthetic_instagram_data(n_users=200, n_posts=150,
hashtags=None, avg_likes=50, avg_comments=5, seed=42):
    random.seed(seed)
    np.random.seed(seed)
    if hashtags is None:
        hashtags = ['tech', 'ai', 'photography', 'travel', 'food']
    users = [random_username(8) for _ in range(n_users)]
    posts = []
    for pid in range(n_posts):
        owner = random.choice(users)
        post_hashtags = list(set(random.choices(hashtags,
k=random.randint(1,3))))
```

```
        like_count = max(0, int(np.random.poisson(avg_likes)))
        comment_count = max(0, int(np.random.poisson(avg_comments)))
        likers = random.choices(users, k=min(len(users), like_count))
        commenters = [random.choice(users) for _ in
range(comment_count)]
        comments = []
        for cid, commenter in enumerate(commenters):
            text_len = random.randint(3,20)
            text = " ".join(random.choices(string.ascii_lowercase,
k=text_len))
            comments.append({'id': f'{pid}_{cid}', 'username':
commenter, 'text': text})
        posts.append({
            'post_id': f'post_{pid}',
            'owner': owner,
            'hashtags': post_hashtags,
            'like_count': len(likers),
            'likers': likers,
            'comments': comments
        })
    return users, posts

def filter_posts_by_hashtag(posts, hashtag):
    return [p for p in posts if hashtag in p['hashtags']]

def build_interaction_graph(posts, include_likes=True,
include_comments=True, like_weight=0.5, comment_weight=1.0):
    G = nx.DiGraph()
    for p in posts:
        owner = p['owner']
        G.add_node(owner)
        if include_likes:
            for liker in p.get('likers', []):
                if liker == owner:
                    continue
                if G.has_edge(liker, owner):
                    G[liker][owner]['weight'] += like_weight
                else:
                    G.add_edge(liker, owner, weight=like_weight,
types=['like'])
        if include_comments:
            for c in p.get('comments', []):
                commenter = c['username']
                if commenter == owner:
                    continue
                if G.has_edge(commenter, owner):
                    G[commenter][owner]['weight'] += comment_weight
                    if 'comment' not in G[commenter][owner]['types']:
```

```
                        G[commenter][owner]['types'].append('comment')
                else:
                    G.add_edge(commenter, owner, weight=comment_weight,
types=['comment'])
    return G

def aggregate_to_undirected(G_dir):
    G = nx.Graph()
    for u, v, data in G_dir.edges(data=True):
        w = data.get('weight', 1.0)
        if G.has_edge(u, v):
            G[u][v]['weight'] += w
        else:
            G.add_edge(u, v, weight=w)
    return G

def compute_metrics(G):
    deg = dict(G.degree(weight='weight'))
    centrality = nx.betweenness_centrality(G, weight='weight')
    clustering = nx.clustering(G, weight='weight')
    top_degree = sorted(deg.items(), key=lambda x: x[1],
reverse=True)[:10]
    top_centrality = sorted(centrality.items(), key=lambda x: x[1],
reverse=True)[:10]
    return {'degree': deg, 'betweenness': centrality, 'clustering':
clustering, 'top_degree': top_degree, 'top_betweenness':
top_centrality}

def detect_communities(G):
    partition = community_louvain.best_partition(G, weight='weight')
    comms = {}
    for node, cid in partition.items():
        comms.setdefault(cid, []).append(node)
    return partition, comms

def draw_graph(G, partition=None, figsize=(12, 8), title='Interaction
Graph'):
    plt.figure(figsize=figsize)
    pos = nx.spring_layout(G, k=0.15, seed=42)
    if partition is None:
        nx.draw_networkx_nodes(G, pos, node_size=100,
node_color='lightblue')
    else:
        cmap = plt.get_cmap('tab20')
        communities = {}
        for n, cid in partition.items():
            communities.setdefault(cid, []).append(n)
        for cid, nodes in communities.items():
```

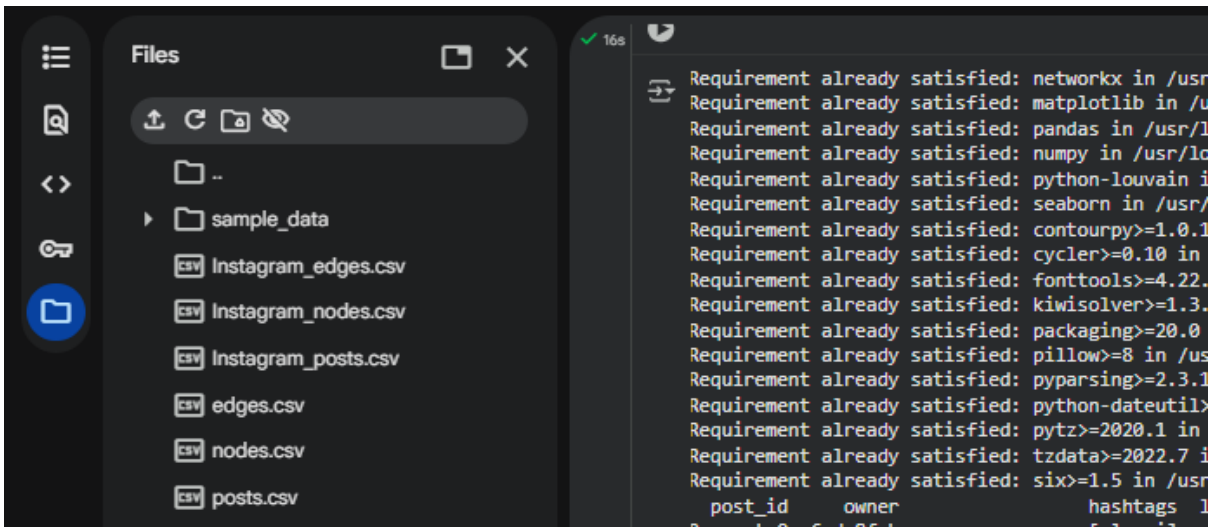
```
            nx.draw_networkx_nodes(G, pos, nodelist=nodes,
node_size=100, node_color=cmap(cid % 20))
    widths = [max(0.1, G[u][v].get('weight',1)*0.3) for u, v in
G.edges()]
    nx.draw_networkx_edges(G, pos, width=widths, alpha=0.6)
    nx.draw_networkx_labels(G, pos, font_size=8)
    plt.title(title)
    plt.axis('off')
    plt.show()

users, posts = generate_synthetic_instagram_data(
    n_users=300, n_posts=250,
hashtags=['tech','ai','datascience','ml','python'], avg_likes=60,
avg_comments=6, seed=123
)

hashtag_to_analyze = 'ai'
filtered_posts = filter_posts_by_hashtag(posts, hashtag_to_analyze)
print(pd.DataFrame(filtered_posts[:5])[['post_id','owner','hashtags','l
ike_count']])

G_dir = build_interaction_graph(filtered_posts, include_likes=True,
include_comments=True, like_weight=0.3, comment_weight=1.0)
G_und = aggregate_to_undirected(G_dir)

metrics = compute_metrics(G_und)
partition, communities = detect_communities(G_und)

all_users = set()
for p in filtered_posts:
    all_users.add(p['owner'])
    all_users.update(p.get('likers', []))
    all_users.update(c['username'] for c in p.get('comments', []))

print('Number of users in filtered posts:', len(all_users))
print('Number of nodes in graph:', G_und.number_of_nodes())
print('Number of edges in graph:', G_und.number_of_edges())

print('Top 10 users by weighted degree:')
print(pd.DataFrame(metrics['top_degree'], columns=['User','Weighted
Degree']))

print('Top 10 users by betweenness centrality:')
print(pd.DataFrame(metrics['top_betweenness'],
columns=['User','Betweenness']))

print('Detected communities (up to 10 shown):')
for cid, members in list(communities.items())[:10]:
```

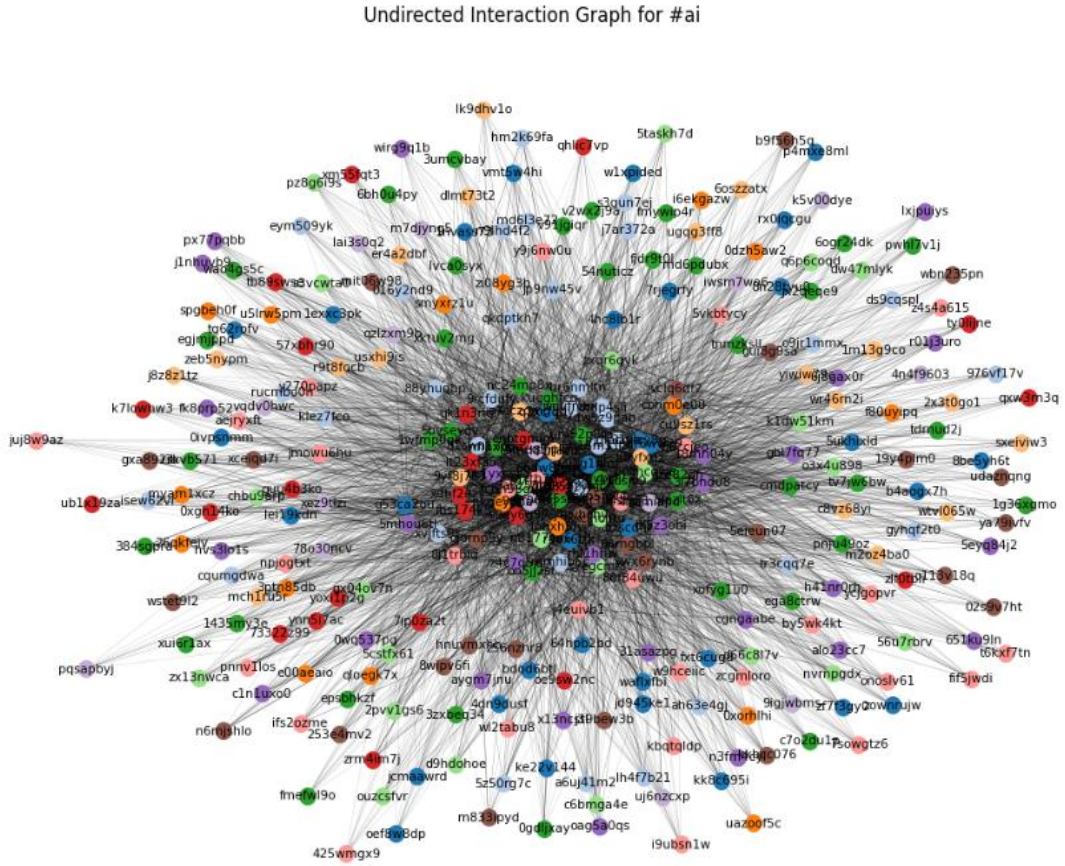
```
    print('community', cid, members[:8])

draw_graph(G_und, partition=partition, figsize=(14,10),
title=f'Undirected Interaction Graph for #{hashtag_to_analyze}')

plt.figure(figsize=(8,4))
degree_values = sorted([d for n, d in G_und.degree(weight='weight')],
reverse=True)
plt.plot(degree_values)
plt.title('Degree Distribution (Weighted)')
plt.xlabel('Rank')
plt.ylabel('Weighted Degree')
plt.show()

plt.figure(figsize=(8,4))
sns.histplot(list(metrics['clustering'].values()), bins=20, kde=True)
plt.title('Clustering Coefficient Distribution')
plt.xlabel('Clustering Coefficient')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(8,4))
sns.scatterplot(x=list(metrics['degree'].values()),
y=list(metrics['betweenness'].values()))
plt.title('Degree vs Betweenness Centrality')
plt.xlabel('Weighted Degree')
plt.ylabel('Betweenness Centrality')
plt.show()

community_sizes = [len(members) for members in communities.values()]
plt.figure(figsize=(6,4))
sns.barplot(x=list(range(len(community_sizes))), y=community_sizes)
plt.title('Community Sizes')
plt.xlabel('Community ID')
plt.ylabel('Number of Users')
plt.show()

nodes_df = pd.DataFrame({'node': list(G_und.nodes())})
edges_df = pd.DataFrame([(u,v,data.get('weight',1.0)) for u,v,data in
G_und.edges(data=True)], columns=['source','target','weight'])
posts_df = pd.DataFrame([{'post_id': p['post_id'], 'owner': p['owner'],
'hashtags': ','.join(p['hashtags']), 'like_count': p['like_count'],
'comment_count': len(p['comments'])} for p in filtered_posts])

nodes_df.to_csv('Instagram_nodes.csv', index=False)
edges_df.to_csv('Instagram_edges.csv', index=False)
posts_df.to_csv('Instagram_posts.csv', index=False)
```

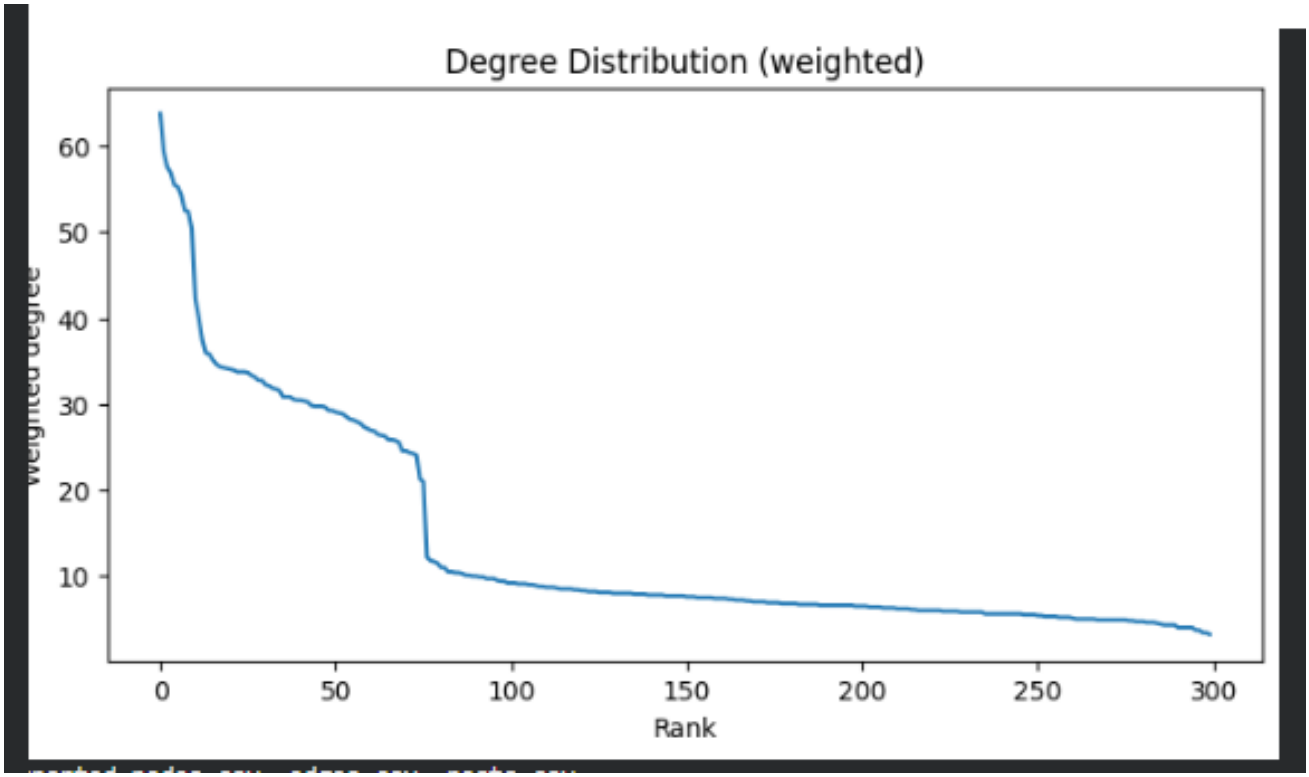
```
print('Exported nodes.csv, edges.csv, posts.csv')
```

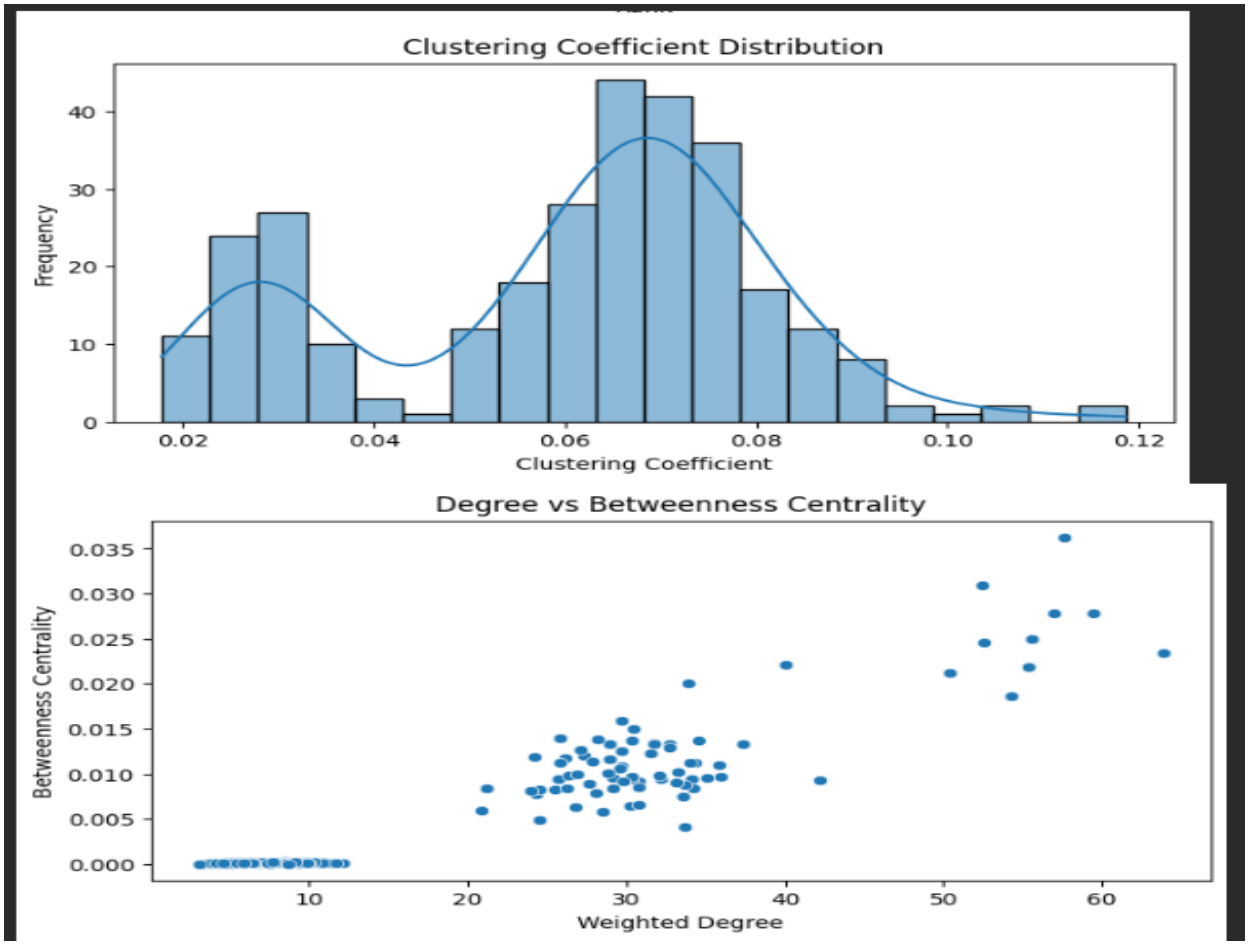
Output:

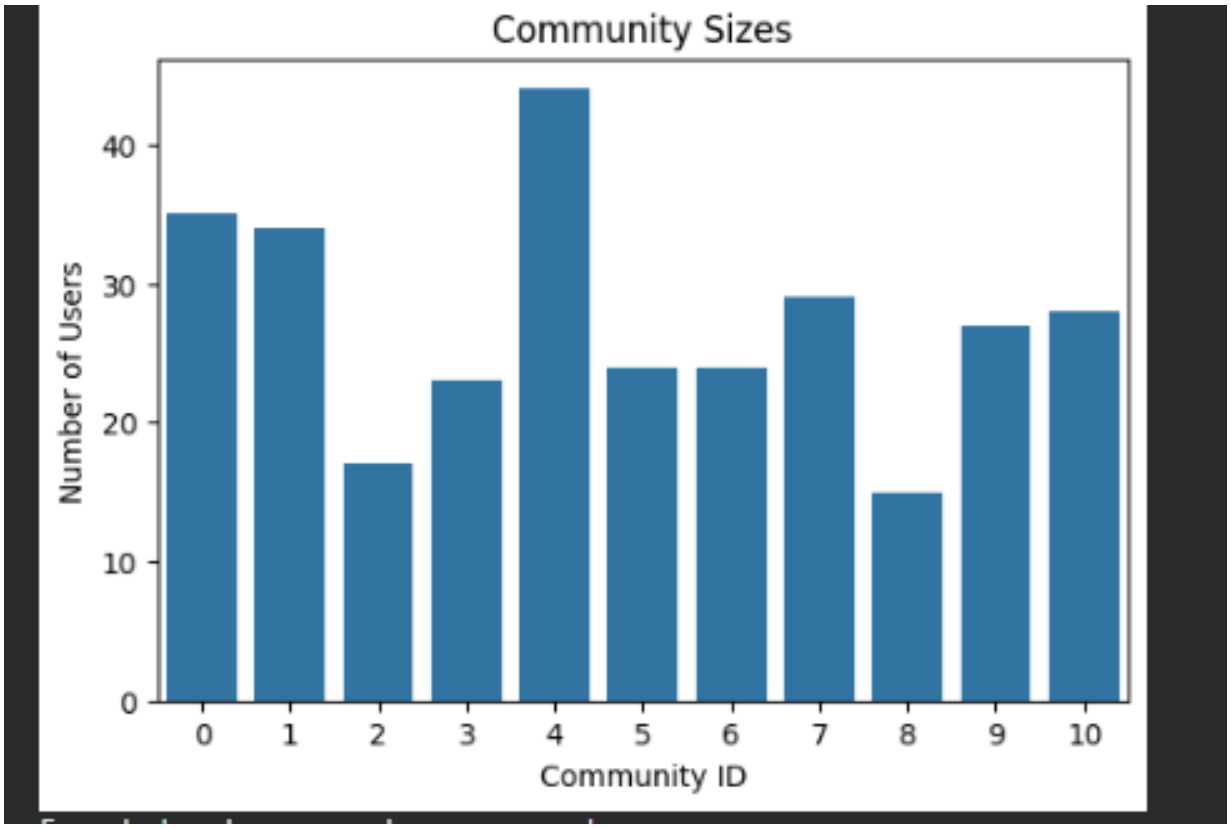
```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
  post_id     owner               hashtags  like_count
0  post_0  6odw8feh               [ml, ai]          65
1  post_2  80f84uwu             [tech, ai]          61
2  post_3  gjornp9y       [datascience, ai]          63
3  post_7  1wfmp0qk                   [ai]          48
4  post_8  8j1trbiq  [datascience, tech, ai]          51
Number of users in filtered posts: 300
Number of nodes in graph: 300
Number of edges in graph: 4805
Top 10 users by weighted degree:
       User  Weighted Degree
0  wtglv2ma            63.8
1  t3u4sm51            59.4
2  pkgey3ha            57.6
3  7fwo5nw4            56.9
4  2bs5sest            55.5
5  vv1g1nlm            55.3
6  7axtb705            54.3
7  v0ey6g9j            52.5
8  tlbyfxdz            52.4
9  31vlcy8g            50.4
Top 10 users by betweenness centrality:
       User  Betweenness
0  pkgey3ha     0.036252
1  tlbyfxdz     0.030953
2  7fwo5nw4     0.027854
3  t3u4sm51     0.027809
4  2bs5sest     0.024941
5  v0ey6g9j     0.024515
6  wtglv2ma     0.023371
7  9a6z53cz     0.022091
8  vv1g1nlm     0.021911
9  31vlcy8g     0.021182
Detected communities (up to 10 shown):
community 0 ['6odw8feh', 'q7xcfqdy', 'vv1g1nlm', 'tttscdqr', '31vlcy8g', 'r4vx6ttk', 'zownrujw', 'g53ca2ou']
community 1 ['jgiachle', '85pzynic', '9rcfdufy', '81swn13b', 'xhdp4sli', 'mrhhib6c', 'xvjfts9t', '88yhuqbp']
community 2 ['pkgey3ha', 'zi08yg3b', 'cbnm0e00', 'smyxrz1u', '3ptn85db', 'lphxh5fv', 'e00aeaio', '0xorhlhi']
community 3 ['9yf8j763', 'tlbyfxdz', 'cu9sz1rs', 'a7cczdmg', 'l8crjacd', 'j8z8z1tz', 'usxhi9is', 'dlmt73t2']
community 4 ['2bs5sest', 'gze82ofi', 'nfihxnuo', 'um92n8ra', '1wfmp0qk', 'on5ii2sf', 'nc24mn8x', '5hysevov']
```




Undirected Interaction Graph for #ai


Degree Distribution (weighted)
Weighted degree
Rank
60
50
40
30
20
10
0
50
100
150
200
250
300


Clustering Coefficient Distribution
Frequency
Clustering Coefficient
0.02
0.04
0.06
0.08
0.10
0.12
0
10
20
30
40
Degree vs Betweenness Centrality
Betweenness Centrality
Weighted Degree
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
10
20
30
40
50
60


Community Sizes
Number of Users
Community ID
0
10
20
30
40
0
1
2
3
4
5
6
7
8
9
10

Questions:

1. How will you authenticate and fetch posts from Instagram?

- I will use OAuth 2.0 authentication through the Instagram Graph API by creating a Facebook App and getting the App ID, App Secret, and Access Token.
- I will fetch posts using API endpoints such as:
- GET /ig_hashtag_search?user_id={user_id}&q={hashtag_name}
- GET /{hashtag_id}/recent_media?user_id={user_id}&fields=id,caption,like_count,comments
- This method will help me securely access Instagram data and retrieve hashtag-based posts along with engagement details like likes and comments.

2. How are edges formed in the interaction graph (likes vs comments)?

- In my interaction graph, each node will represent a user, and edges will represent interactions between users. For example- If any user1 likes or comments on user2's post, I will create an edge between user1 and user2.
- I will give likes a weaker weight and comments a stronger weight, since comments show deeper engagement.

3. State two challenges in real-time data collection from Instagram.

- One major challenge I will face is API Rate Limiting, because Instagram restricts how many requests can be made in a certain time period, which can delay real-time updates.
- Another challenge is Data Sampling/Completeness, since the Hashtag Search API only gives partial data and not every post related to a hashtag.
- Because of these limits, my real-time data might not always be complete or fully up-to-date, which can affect the accuracy of my analysis.

4. How can you identify key influencers on Instagram?

- I will use Degree Centrality to find users who receive the most likes and comments — higher degree means more engagement and popularity.
- I will also apply Betweenness Centrality to find users who act as bridges between different user groups, showing their importance in spreading information.
- To make it more meaningful, I will give comments higher weight than likes, since they represent stronger interaction and real influence.

Practical-2)

Aim: During a tech event, you are tasked with analyzing Reddit community discussions. You must use the Reddit API(PRAW) to collect posts and comments under a specific subreddit and create a discussion network.

Implementation Code:

```
#Practical-2 Reddit Practical
#Reddit Discussion Network

!pip install networkx matplotlib pandas numpy python-louvain seaborn

import random
import string
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
from community import community_louvain

def random_username(length=8):
    return ''.join(random.choices(string.ascii_lowercase +
string.digits, k=length))

def generate_synthetic_reddit_data(n_users=300, n_posts=150,
avg_comments=8, seed=42):
    random.seed(seed)
    np.random.seed(seed)
    users = [random_username(8) for _ in range(n_users)]
    posts = []
    for pid in range(n_posts):
        author = random.choice(users)
        comment_count = max(0, int(np.random.poisson(avg_comments)))
        comments = []
        for cid in range(comment_count):
            commenter = random.choice(users)
            text_len = random.randint(3,20)
            text = " ".join(random.choices(string.ascii_lowercase,
k=text_len))
            parent = author if cid == 0 else random.choice([author] +
[c['username'] for c in comments])
            comments.append({'id': f'{pid}_{cid}', 'username':
commenter, 'text': text, 'reply_to': parent})
```

```
        posts.append({'post_id': f'post_{pid}', 'author': author,
'comments': comments})
    return users, posts

def build_reddit_discussion_graph(posts):
    G = nx.DiGraph()
    for post in posts:
        author = post['author']
        G.add_node(author)
        for comment in post['comments']:
            commenter = comment['username']
            parent = comment['reply_to']
            G.add_node(commenter)
            if commenter == parent:
                continue
            if G.has_edge(commenter, parent):
                G[commenter][parent]['weight'] += 1
            else:
                G.add_edge(commenter, parent, weight=1)
    return G

def compute_metrics(G):
    deg = dict(G.degree(weight='weight'))
    centrality = nx.betweenness_centrality(G, weight='weight')
    clustering = nx.clustering(G.to_undirected(), weight='weight')
    top_degree = sorted(deg.items(), key=lambda x: x[1],
reverse=True)[:10]
    top_centrality = sorted(centrality.items(), key=lambda x: x[1],
reverse=True)[:10]
    return {'degree': deg, 'betweenness': centrality, 'clustering':
clustering, 'top_degree': top_degree, 'top_betweenness':
top_centrality}

def detect_communities(G):
    G_und = nx.Graph()
    for u,v,data in G.edges(data=True):
        G_und.add_edge(u,v,weight=data.get('weight',1))
    partition = community_louvain.best_partition(G_und,
weight='weight')
    comms = {}
    for node, cid in partition.items():
        comms.setdefault(cid, []).append(node)
    return partition, comms

def draw_graph(G, partition=None, figsize=(12,8), title='Reddit
Discussion Graph'):
    plt.figure(figsize=figsize)
    pos = nx.spring_layout(G, k=0.15, seed=42)
```

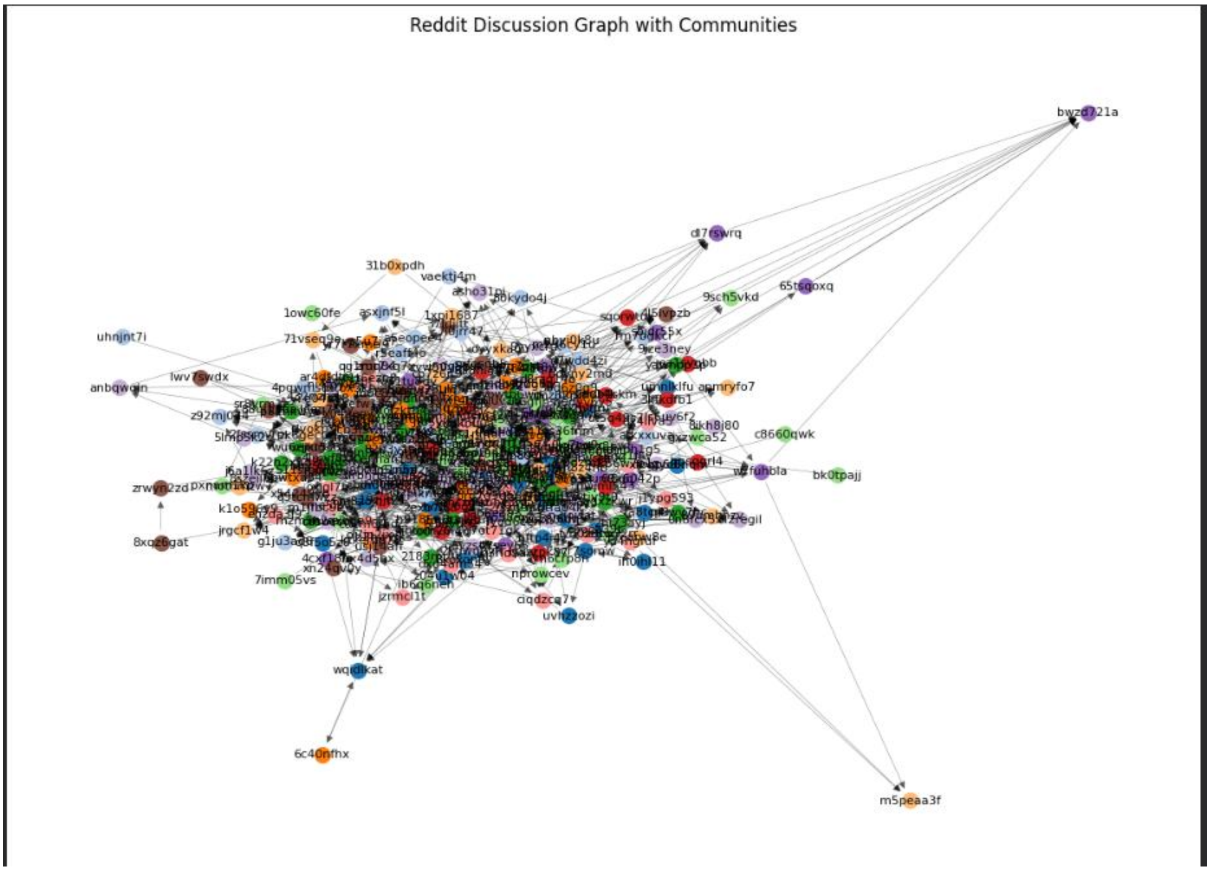
```
    if partition is None:
        nx.draw_networkx_nodes(G, pos, node_size=100,
node_color='lightblue')
    else:
        cmap = plt.get_cmap('tab20')
        communities = {}
        for n, cid in partition.items():
            communities.setdefault(cid, []).append(n)
        for cid, nodes in communities.items():
            nx.draw_networkx_nodes(G, pos, nodelist=nodes,
node_size=100, node_color=cmap(cid % 20))
    widths = [max(0.1, G[u][v].get('weight',1)*0.3) for u, v in
G.edges()]
    nx.draw_networkx_edges(G, pos, width=widths, alpha=0.6)
    nx.draw_networkx_labels(G, pos, font_size=8)
    plt.title(title)
    plt.axis('off')
    plt.show()

users, posts = generate_synthetic_reddit_data(n_users=300, n_posts=180,
avg_comments=7, seed=321)

print("Sample posts (first 5):")
print(pd.DataFrame(posts[:5])[['post_id','author']])

G = build_reddit_discussion_graph(posts)
metrics = compute_metrics(G)
partition, communities = detect_communities(G)

print('Number of users:', len(G.nodes()))
print('Number of edges:', G.number_of_edges())
print('Top 10 users by weighted degree:')
print(pd.DataFrame(metrics['top_degree'], columns=['User','Weighted
Degree']))
print('Top 10 users by betweenness centrality:')
print(pd.DataFrame(metrics['top_betweenness'],
columns=['User','Betweenness']))
print('Detected communities (showing up to 10):')
for cid, members in list(communities.items())[:10]:
    print('community', cid, members[:8])

draw_graph(G, partition=partition, figsize=(14,10), title='Reddit
Discussion Graph with Communities')

plt.figure(figsize=(8,4))
degree_values = sorted([d for n, d in G.degree(weight='weight')],
reverse=True)
plt.plot(degree_values)
```

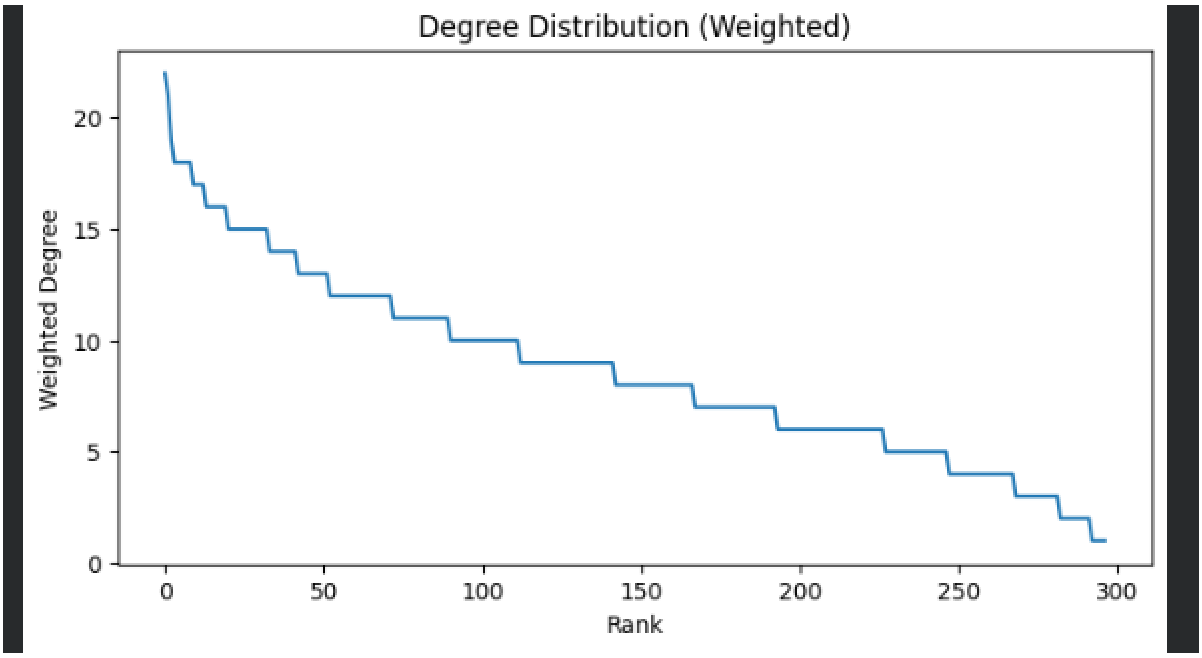
```
plt.title('Degree Distribution (Weighted)')
plt.xlabel('Rank')
plt.ylabel('Weighted Degree')
plt.show()

plt.figure(figsize=(8,4))
sns.histplot(list(metrics['clustering'].values()), bins=20, kde=True)
plt.title('Clustering Coefficient Distribution')
plt.xlabel('Clustering Coefficient')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(8,4))
sns.scatterplot(x=list(metrics['degree'].values()),
y=list(metrics['betweenness'].values()))
plt.title('Degree vs Betweenness Centrality')
plt.xlabel('Weighted Degree')
plt.ylabel('Betweenness Centrality')
plt.show()

community_sizes = [len(members) for members in communities.values()]
plt.figure(figsize=(6,4))
sns.barplot(x=list(range(len(community_sizes))), y=community_sizes)
plt.title('Community Sizes')
plt.xlabel('Community ID')
plt.ylabel('Number of Users')
plt.show()

nodes_df = pd.DataFrame({'node': list(G.nodes())})
edges_df = pd.DataFrame([(u,v,data.get('weight',1)) for u,v,data in
G.edges(data=True)], columns=['source','target','weight'])
posts_df = pd.DataFrame([{'post_id': p['post_id'], 'author':
p['author'], 'comment_count': len(p['comments'])} for p in posts])
nodes_df.to_csv('reddit_nodes_prac2.csv', index=False)
edges_df.to_csv('reddit_edges_prac2.csv', index=False)
posts_df.to_csv('reddit_posts_prac2.csv', index=False)

print('Exported reddit_nodes_prac2.csv, reddit_edges_prac2.csv,
reddit_posts_prac2.csv')
```

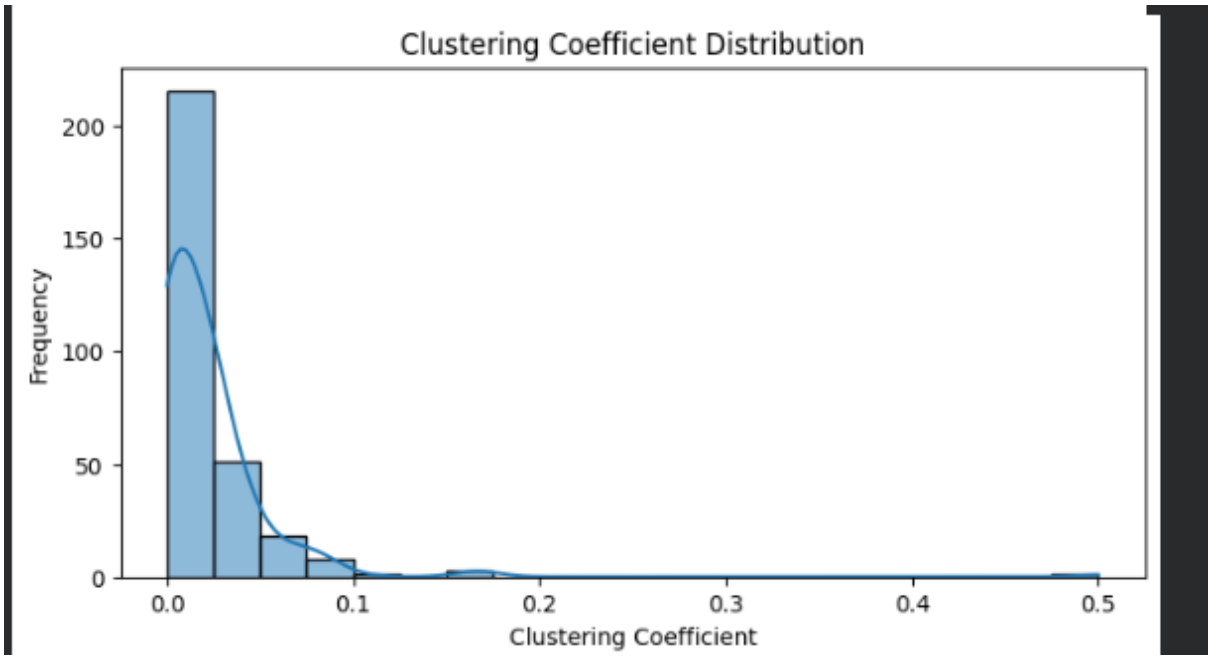
Output:

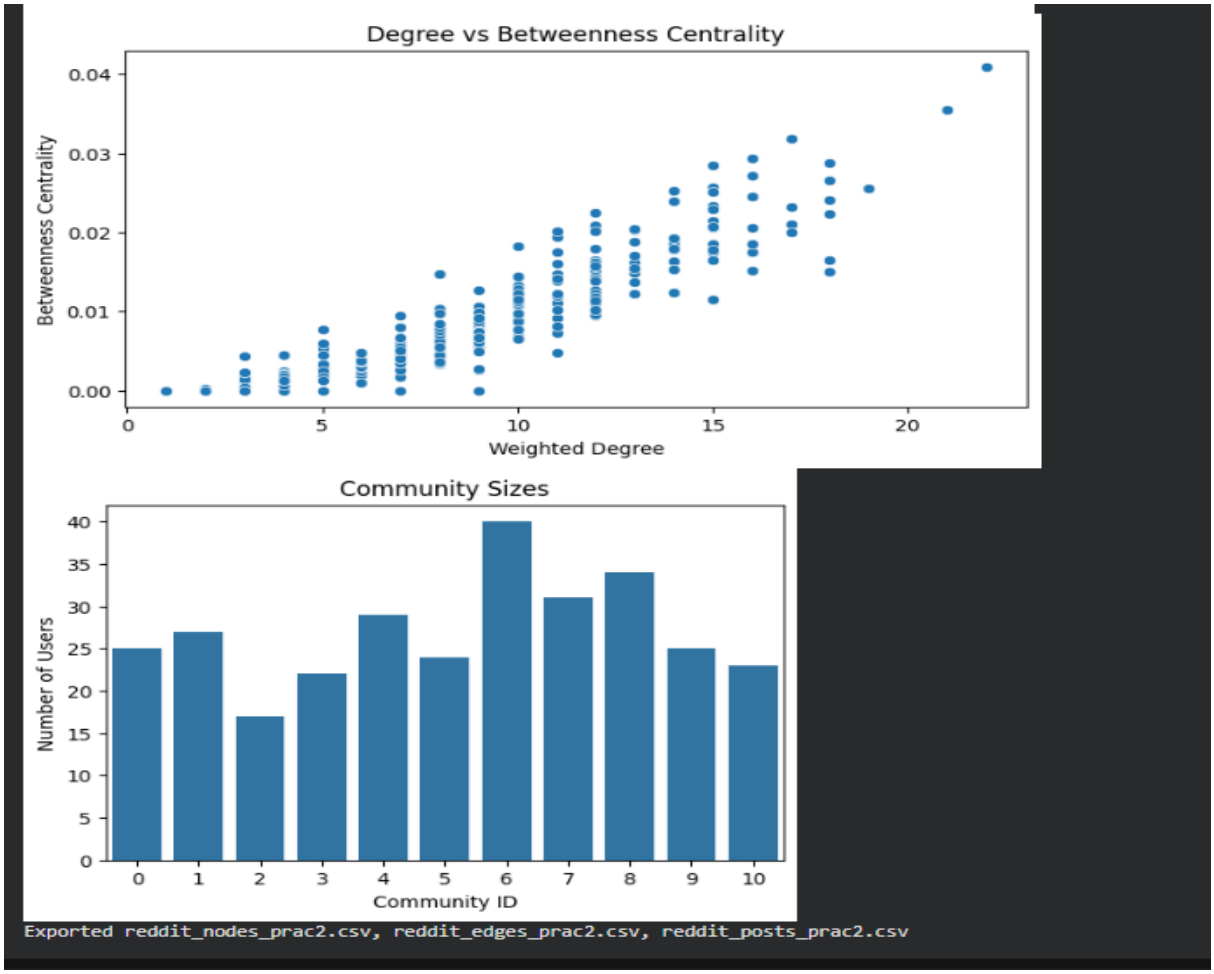
```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Sample posts (first 5):
  post_id    author
0  post_0  61nmi2js
1  post_1  wqidlkat
2  post_2  82u3y1ae
3  post_3  aagxzkwr
4  post_4  anbqwqin
Number of users: 297
Number of edges: 1273
Top 10 users by weighted degree:
       User  Weighted Degree
0  vf5xbbli               22
1  ovyf3ot9               21
2  5wth46k3               19
3  rdzd8071               18
4  6rphgxqg               18
5  zp8we5f8               18
6  cl5k4xg1               18
7  om38nc9w               18
8  ww18cp94               18
9  uy5a6pah               17
Top 10 users by betweenness centrality:
       User  Betweenness
0  vf5xbbli     0.040936
1  ovyf3ot9     0.035511
2  fenczn9z     0.031903
3  zi16kfsd     0.029412
4  zp8we5f8     0.028715
5  738qz6cx     0.028403
6  htpysv9h     0.027188
7  6rphgxqg     0.026574
8  5uep240m     0.025728
```

```
0  vf5xbbli     0.040936
1  ovyf3ot9     0.035511
2  fenczn9z     0.031903
3  zi16kfsd     0.029412
4  zp8we5f8     0.028715
5  738qz6cx     0.028403
6  htpysv9h     0.027188
7  6rphgxqg     0.026574
8  5uep240m     0.025728
9  5wth46k3     0.025593
Detected communities (showing up to 10):
community 0 ['61nmi2js', '2exb7l63', 'wqidlkat', 'o6xnx2xk', '5uep240m', 'c6oca39t', '3jy6dq55', 'ovyf3ot9']
community 1 ['m296cy1u', 'umz62rkf', 'asxjnf5l', 'z92mj074', 'mntfao4k', 'vf5xbbli', 'yrl4o4z7', 'mjg02pvm']
community 8 ['4cxf18io', 'tlksey65', '6j3avtk6', '8ao6e393', '69x6042p', 'bwzd721a', 'a31tu8dy', '65tsqoxq']
community 6 ['7bu8r8h8', '3kfkdfb1', '6aub4skm', 'ro0rym4q', 'fenczn9z', 'bhy9lotu', 'hn3u0ga3', 'sqorwtux']
community 4 ['mzm5hwe2', '2183mrhr', '2hi73ayj', 'jpb4xy8n', 'helnl2it', 'wu6zjtug', 'osd1nmw6', 'p8lmkwy6']
community 2 ['b9185q6t', 'ww18cp94', 'jg0jvgm1', '738qz6cx', 'kvwt4bp6', 'ar4dfdtr', '2effqpkh', 'noizktu4']
community 9 ['amvncti1', '6rphgxqg', 'om38nc9w', 'uy5a6pah', '0chwxrhw', 'emzhals2', 'y27tdn2m', '9jze3ney']
community 7 ['b2hwvfks', 'k2vdud58', 'qz13v9hq', 'ciqdzcq7', 'g7ot71gk', 'rme5lpw3', 'kssoxyqe', 'j1ypg593']
community 5 ['ib6q6neh', '9sch5vkd', 'jm6crp8n', '5wth46k3', 'twssti4r', 'nprowcev', 'i8s5z2qh', 'm2qxoge9']
community 10 ['88ui13ua', 'qg1ruq84', 'usj14aff', 'xn24qv0y', 'rr7wbdqz', 'sw9hfqnx', 'loeq75sy', '1zquuv1b']
/usr/local/lib/python3.12/dist-packages/networkx/drawing/nx_pylab.py:1438: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence,
  node_collection = ax.scatter(
```


Reddit Discussion Graph with Communities
bwzd721a
dl7rswrq
65tsgoxq
9sch5vkd
uhnjnt7i
anbqwqin
c8660qwk
bk0tpajj
8xqz6gat
7imm05vs
wqidkat
6c40nfhx
m5peaa3f
uvhzzozi


Degree Distribution (Weighted)
Weighted Degree
Rank
0
5
10
15
20
0
50
100
150
200
250
300


Clustering Coefficient Distribution
Frequency
Clustering Coefficient
0
50
100
150
200
0.0
0.1
0.2
0.3
0.4
0.5


Degree vs Betweenness Centrality
Betweenness Centrality
Weighted Degree
0.00
0.01
0.02
0.03
0.04
0
5
10
15
20
Community Sizes
Number of Users
Community ID
0
5
10
15
20
25
30
35
40
0
1
2
3
4
5
6
7
8
9
10
Exported reddit_nodes_prac2.csv, reddit_edges_prac2.csv, reddit_posts_prac2.csv

Questions:

1. How will you authenticate and fetch Reddit posts/comments?

- I will use PRAW (Python Reddit API Wrapper) for authentication by registering a Reddit application to get the client ID, client secret, and user agent.
- Using these credentials, I will connect to Reddit's API and fetch data from a specific subreddit using commands like reddit.subreddit('subreddit_name').hot() or .comments().
- This setup allows me to securely access Reddit data and collect both posts and their related comments for further analysis.

2. How are edges formed in the interaction graph (reply vs mention)?

- In my discussion network, nodes will represent Reddit users, and edges will represent interactions between them.
- If user A replies to user B's comment, I will create an edge between A and B. If A mentions B (using "u/username"), that will also form an edge.
- I will assign stronger weights to replies since they represent direct engagement, while mentions will have a lighter weight.

3. State two challenges in real-time Reddit data extraction.

- One challenge is API Rate Limiting, since Reddit's API restricts how many requests can be made per minute, which slows down real-time data collection.
- Another challenge is Dynamic Thread Updates, because Reddit discussions evolve quickly and new comments keep appearing, making it hard to capture a complete snapshot in real time.
- These challenges can cause missing or delayed data, which affects the freshness and accuracy of my discussion analysis.

4. How can you detect the most influential or frequent contributors?

- I will calculate Degree Centrality to identify users who interact the most — those who post or receive many replies are likely key contributors.
- I will also use Betweenness Centrality to find users who connect different conversation threads, showing their influence in bridging discussions.
- By combining these measures, I can highlight users who are either frequent posters or influential participants within the subreddit network.